Application of Mininet

Report of Wireless communication

and mobile network Project

Nie Xiaofang

5110309418

Abstract:

This report is the summary with learning mininet. And it starts with the basics.

In this report I complete some parts of the teacher‘s lab manual.And do something new by myself.

I list all the details of every lab I did.

Experiment 1 Laboratory Setup and Introduction

1. Introduction

This experiment presents a preliminary introduction to the laboratory setup, and is intended to prepare you for a semester full of work. There is basic networking material introduced in this experiment. You may skip through any section that you are comfortable with; however, you are responsible for the contents of this experiment.

Goals of this Assignment

Your goal for this week is to familiarize yourself with the UNIX environment. You will use the UNIX environment to run the commands that we will be using for the remainder of this semester.

Pre-requisites

There are no prerequisites for this lab.

2. Lab Session

Exercise 1: Using the KVM Switch
Create the network in the VM.
terminal> sudo mn --topo single,4 --mac --switch ovsk --controller remote
Open the xterm for host 1.
mininet> xterm h1
Check the switch flow table.
terminal> dpctl dump-flows tcp:127.0.0.1:6634
Start a ping from h1 to h2. If the ping responses are received, then ws1 and ws2 are connected to the network.
mininet> h1 ping -c3 h2
Do you get any replies? Why? Why not?
As you saw before, switch flow table is empty, leading to ping failure.
You'll use dpctl to manually install the necessary flows.
terminal> dpctl add-flow tcp:127.0.0.1:6634 in_port=1,actions=output:2
terminal> dpctl add-flow tcp:127.0.0.1:6634 in_port=2,actions=output:1
This will forward packets coming at port 1 to port 2 and vice-verca. Verify by checking the flow-table
terminal> dpctl dump-flows tcp:127.0.0.1:6634
Run the ping command again.
mininet> h1 ping -c3 h2
If you didn't see any ping replies coming through, it might be the case that the flow-entries expired before you start your ping test. When you do a "dpctl dump-flows" you can see an "idle_timeout" option for each entry, which defaults to 60s. This means that the flow will expire after 60secs if there is no incoming traffic. Run again respecting this limit, or install a flow-entry with longer timeout.
terminal> dpctl add-flow tcp:127.0.0.1:6634 in_port=1,idle_timeout=120,actions=output:2
Besides that, there is no controller connected to the switch and therefore the switch doesn't know

what to do with incoming traffic, leading to ping failure.
Now try to use of_tutorial.py under the directory pox/misc as the controller.
terminal> cd pox
terminal>./pox.py log.level --DEBUG misc.of_tutorial
Run the ping command again.
mininet> h1 ping -c3 h2
Start a ping from h3 to h4. If the ping responses are received, then ws1 and ws2 are connected to the network.

Exercise 2: Using Packet Capture Tools
From h2, send 10 ping messages to h4.
mininet> xterm h2
h2> ping –c10 10.0.0.4
From h3, send 5 ping messages to the loopback interface. Explain what the loopback interface is.
mininet> xterm h3
h3> ping –c5 127.0.0.1
From h3, use tcpdump to monitor all packets with the IP address of h4. Open a new xterm and send 5 ping messages to h4. Explain the meaning of each field in the captured data. (Maybe you need to wait for a while until the result of packet capture shows up)
h3> tcpdump host 10.0.0.4
mininet> xterm h3
h2> ping –c5 10.0.0.4
You can also use tcpdump to monitor all packets with the IP address of h4 in the terminal.
terminal> tcpdump –i s1-eth1 -nn

Experiment 2 Single Segment Networks

1. Introduction

You will get introduced to various issues related to Ethernet networks. To begin with, we will concentrate on simple single-segment networks.

Pre-requisites

You should have finished experiment 1 to get an introduction to the lab setup and the equipment.

Assignment

2. Pre-lab

Learn more about the following UNIX commands: ifconfig, netstat, arp, telnet, ftp. Read about the capture and display filters used in the tcpdump in detail.

Based on your readings, answer the following questions.

Write the command to set the IP address of the interface eth0 to 10.45.25.156/17.

You can find the IP address of the interface by terminal> ifconfig eth0

terminal> sudo ifconfig eth0 10.45.25.156/17

How can you find the MAC address of an interface? Can you use two different commands for this purpose?

terminal> ifconfig –a

You can also find the MAC address of the host which is linked to yourself.

h1> ping –c3 10.0.0.2

h1> arp –a 10.0.0.2

What is the syntax of the tcpdump command for the following:

Capture packets with IP datagrams with source or destination IP address equal to 10.0.0.2

tcpdump ip host 10.0.0.2

Capture packets with IP datagrams with source or destination IP address equal to 10.0.0.2 on interface eth1

tcpdump –i eth1 ip host 10.0.0.2

Capture packets with ICMP messages with source or destination IP address equal to 10.0.0.2

tcpdump host 10.0.0.2 and icmp

Capture packets with TCP segments with source or destination IP address equal to 10.0.0.2

tcpdump tcp host 10.0.0.2

Capture packets with TCP segments with source or destination IP address equal to 10.0.0.2 using port 80

tcpdump tcp port 80 host 10.0.0.2

3. Lab Session

Exercise 1: Configuring IP Addresses in Linux

Create the network in the VM.

Use the ifconfig command to display the configuration of all the interfaces simultaneously. Save the output.

terminal> ifconfig –a

Change the IP address of interface eth0 to 192.168.0.222/24.

terminal> sudo ifconfig eth0 down
terminal> sudo ifconfig eth0 192.168.0.222/24
terminal> sudo ifconfig eth0 up
Use the ifconfig command to display the configuration of all the interfaces and save the output.
Reset the IP address of eth0 to the initial value.
terminal> sudo ifdown eth0
terminal> sudo ifup eth0

Exercise 2: Ethernet Statistics

Use the netstat command to display information about all the network interfaces.
terminal> netstat -i
Use the netstat command to display the IP routing table.
terminal> netstat -r
Use the netstat command to display information about the current TCP and UDP port usage.
terminal> netstat -tcp
terminal> netstat –lt
terminal> netstat -udp
terminal> netstat -lu
Use the netstat command to display the statistics of the networking protocols.
terminal> netstat -s

Exercise 3: Using Packet Filters

Open a xterm on h1. Execute tcpdump with the filter to select all packets with source or destination as 10.0.0.2. From another xterm window, send 5 ping packets to h2.
h1> tcpdump host 10.0.0.2
With the same tcpdump filter as before, ping h3 from h1.
mininet> h1 ping h3
With the same tcpdump filter as before, ping 10.0.2.255 from h1.
mininet> h1 ping 10.0.2.255
Start a tcpdump session to capture all packets on h2. Ping h3 from h1.
h2> tcpdump
Start a tcpdump session to capture all ICMP messages with source or destination as h2 on h3. Ping h2 from h3.
h3> tcpdump host 10.0.0.2 and icmp

Exercise 4: ARP Table and MAC Addresses

View the ARP cache of h2.
h2> arp -a
Delete all the entries from the ARP cache.
h2> arp –d (host)
Start tcpdump on h2 for the address of 10.0.0.4.
h2> tcpdump host 10.0.0.4
Ping h4. Observe the ARP packets in tcpdump. What is the destination address for the ARP

requests?
h2> ping 10.0.0.4
Wait for 5 minutes and view the ARP cache again. Observe that all the entries have been deleted.
h2> arp -a
Create a table with the IP-MAC address mapping of all the connected interfaces. Verify with the two approaches to obtain the MAC addresses.
terminal> ifconfig –a
or
h1> arp –a 10.0.0.2
Start the packet capture with tcpdump on h2. Send 15 ping messages to 10.0.0.200 from h2. Explain the tcpdump output. Observe the frequency and time between the ARP requests.
h2> tcpdump
mininet> h2 ping –c15 10.0.0.200

Exercise 5: IP Address
On ws3, change the IP address of interface eth0 to 10.0.0.1.
h3> ifconfig h3-eth0 10.0.0.1
Delete all the ARP cache entries of h2.
arp –d (host)
Run tcpdump on h2 with capture filter for IP address 10.0.0.1.
h3> tcpdump host 10.0.0.1
Ping h3 from h1.
mininet> h1 ping h3
On h3, change the IP Address of eth0 back to original address.
h3> ifconfig h3-eth0 10.0.0.3
Set the IP addresses as follows
eth0 of h3 as 10.0.0.244 with a netmask of 255.255.255.224
eth0 of h4 as 10.0.0.245 with a netmask of 255.255.255.224
h3> ifconfig h3-eth0 10.0.0.244
h3> ifconfig h3-eth0 netmask 255.255.255.224
h4> ifconfig h4-eth0 10.0.0.245
h4> ifconfig h4-eth0 netmask 255.255.255.224
Run tcpdump at ws1. Capture the following packets in this order:
2 ping messages from ws1 to ws2
2 ping messages from ws1 to ws3
2 ping messages from ws2 to ws3
2 ping messages from ws3 to ws2
2 ping messages from ws3 to ws4
2 ping messages from ws4 to ws3
Save all outputs and the packet capture. Explain the output and the capture.
mininet> h1 ping –c2 h2----------received
mininet> h1 ping –c2 h3----------loss
mininet> h2 ping –c2 h3----------loss
mininet> h3 ping –c2 h2----------Network is unreachable

mininet> h3 ping –c2 h4----------received
mininet> h4 ping –c2 h3----------received
Change all the IP addresses back to the originals, as shown in the figure above.
h3> ifconfig h3-eth0 10.0.0.3
h3> ifconfig h3-eth0 netmask 0.0.0.255
h4> ifconfig h4-eth0 10.0.0.4
h4> ifconfig h4-eth0 netmask 0.0.0.255

Exp3 Lab switching and performance

1. Introduction

In this experiment, we will continue with the exploration of Ethernet networks. Now, we learn how to connect two separate networks together and various issues related to this connection.

Pre-requisites

You should have finished experiment 2 to get an introduction to the Ethernet networks. If you are having any trouble, please get in touch with the TA or the instructor.

2. Assignment

Pre-lab

Learn more about LAN switching at

http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/bridging.htm

http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/transbdg.htm

http://linux-net.osdl.org/index.php/Bridge

Read about netperf at http://www.netperf.org/netperf/training/Netperf.html.

Learn about the usage of network tool traceroute.

Based on your readings, answer the following questions.

1. Explain the difference between hub, bridge/switch and router.
2. Which is more suitable for a network with a high traffic load, hub or switch? Why?

A hub is less expensive, less intelligent, and less complicated of the two. Its job is simple: anything that comes in one port is sent out to the others. PCs connected with a hub can "see" everything transmitted through the hub. A switch does the what a hub does. It "learns" where particular addresses are. It remembers the port that is used for transmission to a certain PC. Later, it knows that this PC is connected to this port and the traffic to this PC needs to only ben sent to that port and not to any of the others. The broadcasting happens at the beginning. Most of the traffic only goes where it needs to go rather than to every port. So a switch is faster. When there's high traffic load, we want to use a switch.

3. What are transparent bridges?

Transparent bridge is a particular case of network bridge. The network bridge simply enables local networks to communicate with each other, but forwards the traffic to all ports. The transparent bridge is capable of redirecting the packets to the proper port, hence it can isolate the networks from broadcast traffic. The transparent bridge keeps a forwarding table that associates addresses to ports. The table is built by learning the network topology from the analysis of the incoming traffic. Transparent bridge is named this way because its presence and operation are transparent to network hosts.

3. Lab Session

[Exercise 1A]

We connected the ethernet interfaces of the PCs. We used the following commands to configure the interfaces of the PCs:

on PC1: `ifconfig eth0 10.0.1.11/24`

on PC2: `ifconfig eth0 10.0.1.21/24`

on PC2: `ifconfig eth1 10.0.1.22/24`

on PC3: `ifconfig eth1 10.0.1.31/24`

We also recorded the MAC addresses on all PCs:

Linux PC	MAC Address of eth0	MAC Address of eth1
PC1	00:02:A5:00:B8:87	00:50:DA:67:BC:95
PC2	00:02:A5:57:1B:8A	00:04:76:CD:61:58
PC3	00:02:A5:17:5D:73	00:10:4B:9B:38:3F
PC4	00:02:A5:02:29:D0	00:10:4B:6F:8D:DB

[Exercise 1B]
We started gbrctl on PC2 using command `gbrct1`. We added a bridge and named it "Bridge 1". We then configured this bridge by adding its two interfaces. For this exercise, we didn't use STP. The STP button was, therefore, disabled. We used command `ifconfig Bridge1 up` to activate the bridge we added in PC2.

[Exercise 1C]
We started the wireshark on PC1's and PC3's interface eth0. We issued ping commands from PC1 to PC2, `ping 10.0.1.21`, and from PC3 to PC2, `ping 10.0.1.22`. They both failed to communicate because once the bridging was activated on PC2, the configured IP addresses should be disabled.

We then cleared the arp tables using command `arp -d` on both PC1 and PC3. We issued a ping command from PC1 to PC3. It went through successfully. We saved the output as ex1c.txt. I also attached the ping staticstics below.

---------- 10.0.1.31 ping statistics ----------
1 packets transmitted, 1 received, 0% packet loss, time 10ms
rtt min/avg/max/mdev = 1.066/1.066/1.066/0.000 ms

Question 4-a, 4-b
PC2 did not modify the source and destination MAC and IP addresses. Attached below is the MAC and IP addresses information in the icmp echo reply message.

---------- icmp echo reply ----------
Destination: HewlettP_00:b8:87 (00:02:a5:00:b8:87)
Source: HewlettP_17:5d:73 (00:02:a5:17:5d:73)
:
Internet Protocol, Src: 10.0.1.31 (10.0.1.31), Dst: 10.0.1.11 (10.0.1.11)

The source and desination MAC and IP addresses in the icmp echo reply packet are shown as above. They are the MAC and IP addresses of PC1 and PC3. PC2 did not show at all.

If PC2 was configured as a router, it would manipulate the MAC address. In the icmp reply packet, it would show PC2's MAC address as the destination MAC address. The source and destination IP addresses would still be PC1 and PC3's.

We issued a traceroute command form PC1 to PC3. It went through successfully. We appended the output in file ex1c.txt. I also attached below the output.

```
----------------------------- traceroute 10.0.1.31 -----------------------------
    traceroute to 10.0.1.31 (10.0.1.31), 30 hops max, 40 byte packets
    1  10.0.1.31 (10.0.1.31)  0.696 ms  0.630 ms  0.542 ms
--------------------------------------------------------------------------------
```

Question 5-a
PC2 is invisible in this saved traceroute output. That is because PC2 was configured as a switch not a router. This is why it is called transparent bridge.

Question 5-b
If PC2 is configured as a IP router, we would see one more entry in the traceroute output indicating PC2.

We changed the IP address of PC3 to 10.0.2.12/24. The previous ping command failed in this case. Because, the bridge thought PC3 was in another network, although they were directly connected, PC2 did not even try to forward the packets.

We save the wireshark captured traffic. Copied below is an icmp echo request packet's source and desitnation MAC and IP addresses information. As we can see the request was from PC1 to PC2. The MAC addresses also indicate PC1 eth0 to PC2 eth0.

```
------------------------------ icmp echo reply ------------------------------
    Destination: HewlettP_57:1b:8a (00:02:a5:57:1b:8a)
    Source: HewlettP_00:b8:87 (00:02:a5:00:b8:87)
    :
    Internet Protocol, Src: 10.0.1.11 (10.0.1.11), Dst: 10.0.1.21 (10.0.1.21)
--------------------------------------------------------------------------------
```

[Exercise 1D]
In this exercise, we had a look at the MAC forwarding table in PC2. The MAC forwarding table for bridge functions as the routing table in a router. We observed two entries:

---------- MAC forwarding table ----------

Port no	Mac address	Is local?	Ageing timer
1	00:02:A5:57:1B:8A	Yes	0.00
2	00:04:76:CD:61:58	Yes	0.00

Both of the two entries were local MAC addresses of PC2. Because the ageing timers were set to zero. The entries other than local MAC addresses were immediately deleted after adding.

We then disabled the bridge on PC2. This was down with two steps. First we used command `ifconfig Bridge1 down` on PC2. Then we seleted delete for Bridge1 in gbrctl. We issued ping commands to verify the deletion.

on PC1: ping -c 1 10.0.1.21
on PC3: ping -c 1 10.0.1.22

The first command went through as we expected. The second one failed. We think this might be a bug in the adding and deleting bridges process. When the bridge was activated, the IP addresses should be disabled as we observed previously. When it was deleted, the IP addresses should come back to working mode. We then issued a ping command from PC1 to PC3 `ping -c 1 10.0.1.31`. This command failed as expected because PC2 was not forwarding packets.

[Exercise 2]
We connected the ethernet interfaces of the PCs and Router1. We used the following commands to configure the interfaces of the PCs:

on PC1: `ifconfig eth0 10.0.1.11/24`
on PC2: `ifconfig eth0 10.0.2.21/24`
on PC3: `ifconfig eth1 10.0.1.31/24`
on PC4: `ifconfig eth1 10.0.1.41/24`

We then used command `show interfaces` on the routers in order to record their MAC addresses on both interfaces.

---------- MAC Adresses of the routers ----------

Router	MAC Address of eth0	MAC Address of eth1
Router1	00:1B:D4:3C:33:10	00:1B:D4:3C:33:11
Router2	00:1B:54:DD:7C:50	00:1B:54:DD:7C:51
Router3	00:1B:D4:DB:9A:70	00:1B:D4:DB:90:71
Router4	00:1B:D4:DB:94:50	00:1B:D4:DB:94:51

We then used the following commands to configure Router1 as a bridge.

Router1

```
yourname# config t
yourname(config)# no ip routing
yourname(config)# bridge 1 protocol ieee
yourname(config)# bridge 1 priority 128
yourname(config)# interface fa0/0
yourname(config-if)# bridge-group 1
yourname(config-if)# bridge-group 1 spanning-disabled
yourname(config-if)# no shutdown
yourname(config-if)# interface fa0/1
yourname(config-if)# brige-group 1
yourname(config-if)# bridge-group 1 spanning-disabled
yourname(config-if)# no shutdown
yourname(config-if)# end
yourname# clear bridge
yourname# clear arp-cache
```

Both interfaces were assigned to the bridge, but the spanning tree protocol is disabled. We used command `arp - a` to verify that the arp caches in PC1 and PC3 were empty. We issued a ping command from PC1 to PC3, `ping -c 1 10.0.1.31` and a traceroute command from PC1 to PC3, `traceroute 10.0.1.31`. We saved the output in file ex2.txt. I also attached the ping statistics and the traceroute output as below.

```
--------------------------- 10.0.1.31 ping statistics ---------------------------
1 packets transmitted, 1 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 1.901/1.901/1.901/0.000 ms

----------------------------- traceroute 10.0.1.31 -----------------------------
[root@PC1 ~]# traceroute 10.0.1.31
traceroute to 10.0.1.31 (10.0.1.31), 30 hops max, 40 byte packets
 1  10.0.1.31 (10.0.1.31)  0.513 ms  0.391 ms  0.337 ms
```

Question5-a

The traceroute output looks just like the traceroute output in Exercise 1C.
Router1 did not appear in the output.

Question 5-b
We couldn't ping Router1 from neither PC1 nor PC3 because Router1 was configured as a transparent bridge. Its configured IP addresses were disabled.

[Exercise 3A]
We connected the hubs and the ethernet interfaces of the PCs. We used the following commands to configure the interfaces of the PCs:

on PC1: `ifconfig eth0 10.0.1.11/24`
on PC2: `ifconfig eth0 10.0.2.21/24`
on PC3: `ifconfig eth1 10.0.1.31/24`
on PC4: `ifconfig eth1 10.0.1.41/24`

We saved the outputs of command `ifconfig -a` on PC1 and PC3 as ex3a2pc1.txt and ex3a2pc3.txt. The initial collsion numbers were zero.

---------- ifconfig -a eth0 output on PC1----------
eth0 Link encap:Ethernet HWaddr 00:02:A5:84:3D:14
inet addr:10.0.1.11 Bcast:10.0.1.255 Mask:255.255.255.0
inet6 addr: fe80::202:a5ff:fe84:3d14/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:759 errors:0 dropped:0 overruns:0 frame:0
TX packets:284 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:73907 (72.1 KiB) TX bytes:18039 (17.6 KiB)

---------- ifconfig -a eth0 output on PC3----------
eth0 Link encap:Ethernet HWaddr 00:02:A5:32:59:E8
inet addr:10.0.1.31 Bcast:10.0.1.255 Mask:255.255.255.0
inet6 addr: fe80::202:a5ff:fe32:59e8/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:507 errors:0 dropped:0 overruns:0 frame:0
TX packets:652 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:55422 (54.1 KiB) TX bytes:46163 (45.0 KiB)

We issued the following
commands to flood the network and hopefully generat some collisions.

on PC1: ping -f 10.0.1.21
on PC3: ping -c 100 10.0.1.41

We saved the outputs of command `ifconfig -a` on PC1 and PC3 again. They are saved as files, ex3a5pc1.txt and ex3a5pc3.txt. We observed 8 collisions on PC3 and 3 collisions on PC1.

```
------------------------ ifconfig -a eth0 output on PC1-------------------------
eth0          Link encap:Ethernet  HWaddr 00:02:A5:84:3D:14
              inet addr:10.0.1.11  Bcast:10.0.1.255  Mask:255.255.255.0
              inet6 addr: fe80::202:a5ff:fe84:3d14/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:625702 errors:0 dropped:0 overruns:0 frame:0
              TX packets:625226 errors:0 dropped:0 overruns:0 carrier:0
              collisions:3 txqueuelen:1000
              RX bytes:61317979 (58.4 MiB)  TX bytes:61261907 (58.4 MiB)

------------------------ ifconfig -a eth0 output on PC3-------------------------
eth0          Link encap:Ethernet  HWaddr 00:02:A5:32:59:E8
              inet addr:10.0.1.31  Bcast:10.0.1.255  Mask:255.255.255.0
              inet6 addr: fe80::202:a5ff:fe32:59e8/64 Scope:Link
              UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
              RX packets:612 errors:0 dropped:0 overruns:0 frame:0
              TX packets:756 errors:0 dropped:0 overruns:0 carrier:0
              collisions:8 txqueuelen:1000
              RX bytes:65522 (63.9 KiB)  TX bytes:56131 (54.8 KiB)
------------------------------------------------------------------------------
```

[Exercise 3B]
We replaced the hubs with one switch. We repeated the previous exercise. The outputs of command `ifconfig -a` on PC1 and PC3 were saved as files, ex3b2pc1.txt and ex3b2pc3.txt. The collisions for this exercise started as 8 for PC3 and 3 for PC1. After the network flood using the two commands list above, we recorded the outputs of command `ifconfig -a` on both PC1 and PC3 again. This time we didn't observe any more collision generated.

(For Exercise 3, I believe something is wrong with the file we saved. I remembered observing those numbers of collisions in the lab and recorded them, but the file we saved indicates that the collision for PC1 eth0 didn't change after Exercise 3B and the collision of PC3 changed back to zero after Exercise 3B.)

[Exercise 4A]

We connected the ethernet interfaces of the PCs and routers. We used the default IP addresses for the PCs this time, to save us the step of configuring the interfaces. We used the following commands on all routers to configure the them as bridges.

Router#

```
yourname# config t
yourname(config)# no ip routing
yourname(config)# bridge 1 protocol ieee
yourname(config)# bridge 1 priority 128
yourname(config)# interface fa0/0
yourname(config-if)# bridge-group 1
yourname(config-if)# bridge-group 1 spanning-disabled
yourname(config-if)# no shutdown
yourname(config-if)# interface fa0/1
yourname(config-if)# brige-group 1
yourname(config-if)# bridge-group 1 spanning-disabled
yourname(config-if)# no shutdown
yourname(config-if)# end
yourname# clear bridge
yourname# clear arp-cache
```

We connected PC2 with the real hub. For this exercise PC2 was configured as a host not a bridge. We verified that all arp caches on PC1, PC2, and PC3 were empty. We issued the following ping commands and saved the output of `show bridge` on each router after each ping commnd.

on PC1: ping -c 1 10.0.1.12
on PC2: ping -c 1 10.0.1.11
on PC2: ping -c 1 10.0.1.14
on PC3: ping -c 1 10.0.1.12

We observed the learning procedure of the bridges. We used Router1 as an example. Copied below are the outputs of `show bridge` on Router1 after the first, second, and third ping commands.

---------- after ping 1: Router1 ----------
yourname#show bridge

Total of 300 station blocks, 296 free
Codes: P - permanent, S - self

Bridge Group 1:

Address	Action	Interface	Age	RX count	TX count
001b.54dd.7c50	forward	FastEthernet0/1	4	1	0
0002.a517.5d73	forward	FastEthernet0/1	3	2	0
0002.a500.b887	forward	FastEthernet0/0	0	3	3
0002.a557.1b8a	forward	FastEthernet0/1	0	8	6

---------- after ping 2: Router1 ----------
yourname#show bridge

Total of 300 station blocks, 298 free
Codes: P - permanent, S - self

Bridge Group 1:

Address	Action	Interface	Age	RX count	TX count
0002.a500.b887	forward	FastEthernet0/0	0	6	6
0002.a557.1b8a	forward	FastEthernet0/1	0	11	9

---------- after ping 3: Router1 ----------
yourname#show bridge

Total of 300 station blocks, 296 free
Codes: P - permanent, S - self

Bridge Group 1:

Address	Action	Interface	Age	RX count	TX count
0002.a500.b887	forward	FastEthernet0/0	1	6	6
0002.a557.1b8a	forward	FastEthernet0/1	1	13	9
001b.d4db.9a70	forward	FastEthernet0/1	0	1	0
0002.a502.29d0	forward	FastEthernet0/1	0	0	0

We can see that before the first ping command, Router1 has MAC address entires for Router2 eth0, PC1 eth0, PC2 eth0, and PC3 eth0. After the second ping command, Router1 has MAC address entires for PC1 eth0 and PC2 eth0. After the third ping command Router1 has MAC address entires for PC1 eth0, PC2 eth0, PC4 eth0, and Router3 eth0.

The first ping command was from PC1 to PC2, the entries for PC3 eth0 and Router2 eth0 were not used. They were, therefore deleted, after the ageing timer expired. After the thir ping command, Router1 learned about PC4 eth0 and Router3 eth0, and added them in the MAC forwarding table. If there would be packets to those two MAC addresses, based on the mapping, Router1 would know to

send them through its interface eth1 instead of broadcasting.

[Exercise 4B]
In this exercise we learned about how do changing of a location affects the bridge and the hub. We first issued some ping commands to help the bridges to create or refresh their MAC forwarding tables. We then disconnect PC2 with its hub and connected it to the switch of PC4.

We should've observed some failures of the pinging because PC2 was connected with a hub, which did not learn about the moving of PC2. Router2 had no way not know that PC2's MAC address should be added in its MAC forwarding table matching with the interface eth0. Eventually old enties would be deleted and the icmp echo requests would flood on all ports. New connections would be built.

However, we didn't observe as discribed because there was another machanism that helped PC2 to broadcast its new conncetion by itself. We didn't observe any unsuccessful pinging.

We repeated this exercise with PC3. PC3 was connected with a switch. The bridge Router3 should learn about this change and delete the entry of PC3's MAC address mapped with its interface eth0 while add an entry of PC3's MAC address mapped with its interface eth1.

Exp4 Dynamic Routing

1. Introduction

In this experiment, you will learn how to configure the network devices to support dynamic routing protocols. You will see the operation of the dynamic routing algorithms and how they update according to the network topology. These protocols are used in large networks.

2. Assignment

Pre-lab

Learn more about the dynamic routing protocols RIP, OSPF and BGP by following links from http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/index.htm. Read about the triggered updates and hold-down mechanisms for solving the count to infinity problem. Also, learn about Zebra at http://www.zebra.org/.

[Question 1]

Provide the command that configures a Linux PC as an IP router.

On a Linux system, IP forwarding is enabled when the file /proc/sys/net/ipv4/ip_forward contains a 1 and disabled when it contains a 0. This change is not permanent, however, it will be lost when the system is rebooted. The command to make this change is:

on PC# `echo "1" > /proc/sys/net/ipv4/ip_forward`

[Question 2]

What are the main differences between a distance vector routing protocol and a link state routing protocol? Give examples for each type of protocol.

Routing Protocols allow routers to dynamically advertise and learn routes, determine which routes are available and which are the most efficient routes to a destination. There are two major types of algorithms for IP routing: Distance Vector Routing and Link State Routing.

Basically, Distance Vector protocols determine the best path on how far the destination is, while Link State protocols are capable of using more sophisticated methods taking into consideration link variables, such as bandwidth, delay, reliability and load. How does a router determine wheter datagrams to a particular host can be directly delivered through one of its interfaces?

The IP Distance Vector routing protocols still in use today are: Routing Information Protocol (RIP v1 and v2) and Interior Gateway Routing Protocol

OSPF, IS-IS and EIGRP are some of the link-state routing protocols in use today.

[Question 3]
What are the differences between an intradomain routing protocol (also called interior gateway protocol, or IGP) and an interdomain routing protocol (also called exterior gateway protocol, or EGP)? Give examples for each type of protocol.

IGP is used within a single autonomous system. It has a single network administration. The nodes, networks share the unique routing policy. IGP makes best use of network resources. The interior gateway protocols can be divided into two categories: 1) Distance-vector routing protocol and 2) Link-state routing protocol. Examples provided in Question 2 such as RIP, IGRP, OSPF are IGP protocols.

EGP is used among different autonomous systems. The networks in the system have their independent administrative entities. The communication is between independent network infrastructures. Border Gateway Protocol (BGP) is a recent exterior gateway protocol.

[Question 4]
Which routing protocls are supported by the software package Zebra?

Zebra is a routing software package that provides TCP/IP based routing services with routing protocols support such as RIP, OSPF and BGP. In addition to traditional IPv4 routing protocols, Zebra also supports IPv6 routing protocols. Zebra also supports special BGP Route Reflector and Route Server behavior.

[Question 5]
In the Zebra software package, the processes ripd, ospfd, and bgpd deal, respectively, with the routing protocls RIP, OSPF, and BGP. Which role does the process zebra play?

The process zebra changes the kernel routing table and redistributes routes between different routing protocols. The ripd daemon handles the RIP protocol, while ospfd is a daemon which supports OSPF version 2. bgpd supports the BGP-4 protocol.

[Question 6]
Describe how a Linux user accesses the processes of Zebra (zebra, ripd, ospfd, bgpd) to configure routing algorithm parameters?

Each daemon has it's own configuration file and terminal interface. When we configure a static route, it must be done in zebra configuration file. When we configure BGP network it must be done in bgpd configuration file. To resolve the problem, Zebra provides integrated user interface shell called vtysh. vtysh connects to each daemon with UNIX domain socket and then works as a proxy for user input.

[Question 7]
What is the main differece between RIP version 1 (RIPv1) and RIP version 2 (RIPv2)?

The main difference between RIPv1 and RIPv2 is that RIPv2 enables the use of a simple authentication mechanism to secure table updates. More importantly, RIPv2 supports subnet masks, a critical feature that is not available in RIPv1. To describe the difference in more detail they are:

1. RIPv2 allows for password authentication of the routing table updates between routers
2. RIPv2 allows for the use of a 16 bit "route tag" that can identify individual routes and imported routes (from other protocols), or be used in other ways by implementors of the protocol.
3. In subnetted IP networks, routers running RIPv1 cannot determine the configured subnet mask (as opposed to the "native" or "natural" one) because RIPv1 messages do not carry that information. RIPv2 rectifies this situation by specifying a 32 bit subnet mask field.
4. RIPv2 has a "next hop" field that can identifiy another router on the local network as the best next hop in the path to the destination being advertised. Specifying a value of 0.0.0.0 in this field indicates that routing should be via the originator of the RIP advertisement, the normal behavior for RIPv1.

[Question 8]
Explain what it means to run RIP in passive mode.

Passive routers listen and update their routes based on advertisements, but do not advertise; active routers advertise their routes (reachability information) to others. Typically, routers run RIP in active mode, while hosts use passive mode.

[Question 9]

Explain the meaning of triggered updates in RIP.

Triggered updates allow a RIP router to announce changes in metric values almost immediately rather than waiting for the next periodic announcement. The trigger is a change to a metric in an entry in the routing table. For example, networks that become unavailable can be announced with a hop count of 16 through a triggered update. Note that the update is sent almost immediately , where a time interval to wait is typically specified on the router. If triggered updates were sent by all routers immediately, each triggered update could cause a cascade of broadcast traffic across the IP internetwork.

Triggered updates improve the convergence time of RIP internetworks but at the expense of additional broadcast traffic as the triggered updates are propagated.

[Question 10]
Explain the concept of split-horizon in RIP.

Split horizon helps reduce convergence time by not allowing routers to advertise networks in the direction from which those networks were learned. The only information sent in RIP announcements are for those networks that are beyond the neighboring router in the opposite direction. Networks learned from the neighboring router are not included.

Split horizon eliminates count-to-infinity and routing loops during convergence in single-path internetworks and reduces the chances of count-to-infinity in multi-path internetworks.

[Question 11]
What is an autonomous system (AS)? Which roles do autonomous systems play in the internet?

Autonomous system (AS) is a collection of connected IP routing prefixes under the control of one or more network operators that presents a common, clearly defined routing policy to the Internet. Each autonomous system is managed independently with respect to BGP

[Question 12]
What is the AS number of you institution? Which autonomous system has AS number 1?

University of hawaii's AS number is: AS6360. I found it by searching "whois

hawaii.edu", which gave me a page: http://cqcounter.com/whois/?query=hawaii.edu

The AS # 1 is Level 3 Communications, Inc. 1025 Eldorado Blvd.
I found it at a online AS number loopup: http://enc.com.au/itools/aut-num.php

[Question 13]
Explain the terms stub AS, multihomed AS, and transit AS?

A stub Autonomous System refers to an AS that is connected to only one other AS.

A multihomed Autonomous System is an AS that maintains connections to more than one other AS. This allows the AS to remain connected to the Internet in the event of a complete failure of one of their connections. However, this type of AS would not allow traffic from one AS to pass through on its way to another AS.

A transit Autonomous System is an AS that provides connections through itself to other networks. That is, network A can use network B, the transit AS, to connect to network C.

3. Lab Session

[Exercise 1A]
We connected the ethernet interfaces of the PCs and the routers. We used the following commands to configure the IP addresses and the interfaces on the PCs:

on PC1: `ifconfig eth0 10.0.1.10/24`
on PC2: `ifconfig eth0 10.0.2.10/24`
on PC3: `ifconfig eth0 10.0.3.10/24`
on PC4: `ifconfig eth0 10.0.4.10/24`

We used the following commands to turn on the RIP protocol and configure the IP addresses and interfaces on the routers:

<Router1>
on PC1: `minicom`
username: cisco
password: cisco
yourname# config t
yourname(config)# no ip routing
yourname(config)# ip routing
yourname(config)# router rip
yourname(config-router)# version 2
yourname(config-router)# network 10.0.0.0
yourname(config-router)# interface FastEthernet0/0
yourname(config-if)# no shutdown

```
yourname(config-if)# ip address 10.0.1.1 255.255.255.0
yourname(config-if)# interface FastEthernet0/1
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.2.1 255.255.255.0
yourname(config-if)# end
yourname# clear ip route *
```

<Router2>

```
yourname(config)# router rip
yourname(config-router)# version 2
yourname(config-router)# network 10.0.0.0
yourname(config-router)# interface FastEthernet0/0
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.2.2 255.255.255.0
yourname(config-if)# interface FastEthernet0/1
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.3.2 255.255.255.0
yourname(config-if)# end
yourname# clear ip route *
```

<Router3>

```
yourname(config)# router rip
yourname(config-router)# version 2
yourname(config-router)# network 10.0.0.0
yourname(config-router)# interface FastEthernet0/0
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.3.3 255.255.255.0
yourname(config-if)# interface FastEthernet0/1
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.4.3 255.255.255.0
yourname(config-if)# end
yourname# clear ip route *
```

We used command `show ip route` to display the routing tables on every router to ensure the configuration was correct. We observed four entries in each routing table. Since the RIP routing protocol was turned on, the routers exchanged routing information with their directly-connected neighbours. When the routing tables stablized, each router had information about all networks in the system.

We issued commands to start the zebra and the ripd processes on all four PCs. We also connected to the ripd processes via Telnet. The commands we used were:

```
on PC#: `/etc/rc.d/init.d/zebra start`
on PC#: `/etc/rc.d/init.d/ripd start`
```

on PC#: `telnet localhost 2602`

After entering the password, the prompt symbol became "ripd#". In order to configure the hosts and set them to run RIP in passive mode, we used commands:

on PC#: ripd# `config t`
on PC#: ripd(config)# `router rip`
on PC#: ripd(config-router)# `version 2`
on PC#: ripd(config-router)# `network 10.0.0.0/8`
on PC#: ripd(config-router)# `passive-interface eth0`
on PC#: ripd(config-router)# `end`

We used command `netstat -rn` to display the routing tables on all four PCs. We saved the output of this command on each PC as ex3a2pc1.txt, ex3a2pc2.txt, ex3a2pc3.txt, and ex3a2pc4.txt.

<PC1 routing table>
Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.4.0	10.0.1.1	255.255.255.0	UG	0	0	0	eth0
10.0.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.2.0	10.0.1.1	255.255.255.0	UG	0	0	0	eth0
10.0.3.0	10.0.1.1	255.255.255.0	UG	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

<PC2 routing table>
Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.4.0	10.0.2.2	255.255.255.0	UG	0	0	0	eth0
10.0.1.0	10.0.2.1	255.255.255.0	UG	0	0	0	eth0
10.0.2.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.3.0	10.0.2.2	255.255.255.0	UG	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

<PC3 routing table>
Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.4.0	10.0.3.3	255.255.255.0	UG	0	0	0	eth0
10.0.1.0	10.0.3.2	255.255.255.0	UG	0	0	0	eth0
10.0.2.0	10.0.3.2	255.255.255.0	UG	0	0	0	eth0
10.0.3.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

<PC4 routing talbe>

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.4.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.1.0	10.0.4.3	255.255.255.0	UG	0	0	0	eth0
10.0.2.0	10.0.4.3	255.255.255.0	UG	0	0	0	eth0
10.0.3.0	10.0.4.3	255.255.255.0	UG	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

We connected Router4 with network 10.0.4.0 and 10.0.2.0. We used the following commands to configure Router4's ip addresses on two of its interfaces. We also configured Router4 to run RIP.

<Router4>

```
on PC1: `minicom`
username: cisco
password: cisco
yourname# config t
yourname(config)# no ip routing
yourname(config)# ip routing
yourname(config)# router rip
yourname(config-router)# version 2
yourname(config-router)# network 10.0.0.0
yourname(config-router)# interface FastEthernet0/0
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.4.4 255.255.255.0
yourname(config-if)# interface FastEthernet0/1
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.2.4 255.255.255.0
yourname(config-if)# end
yourname# clear ip route *
```

We used the command `netstat -rn` to display the routing tables on the PCs. We observed differences comparing to the routing tables before connecting Router4. We saved the converged routing tables for all PCs as ex3a5pc1.txt, ex3a5pc2.txt, and ex3a5pc3.txt.

<PC1 routing table>

Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.4.0	10.0.1.1	255.255.255.0	UG	0	0	0	eth0
10.0.1.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.2.0	10.0.1.1	255.255.255.0	UG	0	0	0	eth0
10.0.3.0	10.0.1.1	255.255.255.0	UG	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

<PC2 routing table>
Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.4.0	10.0.2.4	255.255.255.0	UG	0	0	0	eth0
10.0.1.0	10.0.2.1	255.255.255.0	UG	0	0	0	eth0
10.0.2.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.3.0	10.0.2.2	255.255.255.0	UG	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

<PC3 routing table>
Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.4.0	10.0.3.3	255.255.255.0	UG	0	0	0	eth0
10.0.1.0	10.0.3.2	255.255.255.0	UG	0	0	0	eth0
10.0.2.0	10.0.3.2	255.255.255.0	UG	0	0	0	eth0
10.0.3.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

<PC4 routing table>
Kernel IP routing table

Destination	Gateway	Genmask	Flags	MSS	Window	irtt	Iface
10.0.4.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
10.0.1.0	10.0.4.4	255.255.255.0	UG	0	0	0	eth0
10.0.2.0	10.0.4.4	255.255.255.0	UG	0	0	0	eth0
10.0.3.0	10.0.4.3	255.255.255.0	UG	0	0	0	eth0
192.168.122.0	0.0.0.0	255.255.255.0	U	0	0	0	virbr0

From the routing table comparson, PC2 and PC4 updated their routing tables. PC2 changed to use Router4, gateway 10.0.2.4, for packets with a destination of network 10.0.4.0. PC4 also changed to use Router4, gateway 10.0.4.4, for packets with a destination of network 10.0.2.0 or network 10.0.1.0. The communication between PC2 and PC4 changed to go through Router4 instead of using Router3 plus Router2. The communication between PC1 and PC4 changed to use Router4 plus Router1 instead of using Router3 plus Router2 plus Router1.

It took very little time for the system to update its routing tables. We oberserved differences almost instantaneously. RIP routers maintain only the best route (the route with the lowest metric value) to a destination. After updating its routing table, the router immediately begins transmitting routing updates to inform other network routers of the change. These updates are sent independently of the regularly scheduled updates that RIP routers send.

[Exercise 1B]
We issued a ping command from PC4 to PC1 `ping 10.0.1.10` with no time limitation. Then, we disconnected the ethernet cable connected to interface eth0/0 on Router4. We observed the "Destination Host Unreachable" messages.

A couple of minutes later, the ping command was successful again. We started to observe ICMP echo reply messages again "icmp_seq=# ttl=61 time=#.# ms" at PC4. This indicated that an alternate path had been found between PC4 and PC1. We stopped the ping command with Ctrl-C and saved the ping messages and the ping statistics as ex3b4pingpc4.txt.

---------- 10.0.1.10 ping statistics ----------
229 packets transmitted, 26 received, +117 errors, 88% packet loss, time 228973ms rtt min/avg/max/mdev = 0.537/0.760/2.144/0.388 ms, pipe 4

There were 229 packets transmitted and 26 received. So there were 203 packets lost. The time interval of each packet was 1 sec. So it took 3 min 26 sec to reestablish the connection. So RIP took 3 min 26 sec to update the routing tables.

[Exercise 2A]
We used a ethernet cable to connect the eth1 of PC3 to the switch that was connected to PC1. We used command `ifconfig eth1 10.0.1.11/24` to configure the eth1 on PC3.

We issued commands to start the zebra and the ripd processes on PC3. We also enabled the IP forwarding on PC3. We then connected to the ripd processes via Telnet. The commands we used were:

on PC3: `/etc/rc.d/init.d/zebra restart`
on PC3: `/etc/rc.d/init.d/ripd restart`
on PC3: `echo "1" > /proc/sys/net/ipv4/ip_forward`
on PC3: `telnet localhost 2602`

We issued the following commands to enable RIP on both interfaces of PC3. Since PC3 was set up as an IP router, the interfaces were enabled in the active mode.

<PC3>
on PC3: `enable`
on PC3: ripd# `config t`
on PC3: ripd(config)# `router rip`
on PC3: ripd(config-router)# `version 2`

```
on PC3: ripd(config-router)# `network 10.0.0.0/8`
on PC3: ripd(config-router)# `no passive-interface eth0`
on PC3: ripd(config-router)# `no passive-interface eth1`
on PC3: ripd(config-router)# `redistribute connected`
on PC3: ripd(config-router)# `end`
```

We started wireshark on both interfaces of PC3. We set a display filter, rip. Then we issued command `netstat -rn` on PC1 and PC4 to observe the changes of the routing tables.

We observed that PC1 changed to use PC3, gateway 10.0.1.11, to get to network 10.0.3.0 instead of using Router 1 as the gateway the get to network 10.0.3.0. PC4 also changed to go through Router3 plus PC3 to get to network 10.0.1.0 instead of using Router3 plus Router2 plus Router1 to get to network 10.0.1.0.

[Exercise 2B]
We used the following command to configure the routers. We set the hold-down timer to 0 and disabled the triggered updates on all routers.

```
on PC1: `minicom`
username: cisco
password: cisco
yourname# config t
yourname(config)# ip routing
yourname(config)# router rip
yourname(config-router)# version 2
yourname(config-router)# network 10.0.0.0
yourname(config-router)# timers basic 30 180 0 240
yourname(config-router)# flash-update-threshold 30
yourname(config-if)# end
yourname# clear ip route *
```

We also used the following commands to make Router1 unattractive by increasing the cost metric of both interfaces at Router1 from 1 to 10.

```
yourname# config t
yourname(config)# router rip
yourname(config-router)# offset-list 0 out 10 eth0/0
yourname(config-router)# offset-list 0 out 10 eth0/1
yourname(config-if)# end
yourname# clear ip route *
```

We confirmed the configuration by issuing a traceroute from PC1 to PC4. We

observed that all packets from PC1, network 10.0.1.0, went through PC3 instead of Router1.

We started wireshark on PC3 interface eth0. We set the display filter to be rip. Then we issued a ping command from PC2 to PC1, `ping 10.0.1.10`, with no time limitation. We also enabled the debugging mode of RIP on Router3. This was done to have Router3 display all received RIP packets. We used command `debug ip rip` on Router3.

Then, we disabled the connection between PC3 interface eth1 and network 10.0.1.0 using command `ifconfig eth1 down` on PC3. We obsvered the "Destination Host Unreachable" messages and "Network is unreachable" messages. After several minutes, we started to observe ICMP echo replay packets "icmp_seq=# ttl=61 time=#.# ms" again. We calculated the time for RIP to reestablish the routing tables. The time was 2 min 49 sec.

We saved the debug messages on Router3 as ex4b9debug.txt. We also saved the RIP packets that explained the count-to-infinity problem as ex4b10pc3.txt. We saved all packets with detailed information. I cleaned it up and saved the six packets that explained the problem as ex4b10wireshark.txt.

```
No.     Time           Source                Destination           Protocol Info
      66 58.801925      10.0.3.10             224.0.0.9             RIPv2    Response
 :
 (sniped)
 :
 Routing Information Protocol
     Command: Response (2)
     Version: RIPv2 (2)
     Routing Domain: 0
     IP Address: 10.0.1.0, Metric: 1
         Address Family: IP (2)
         Route Tag: 0
         IP Address: 10.0.1.0 (10.0.1.0)
         Netmask: 255.255.255.0 (255.255.255.0)
         Next Hop: 0.0.0.0 (0.0.0.0)
         Metric: 1
     IP Address: 192.168.122.0, Metric: 1
         Address Family: IP (2)
         Route Tag: 0
         IP Address: 192.168.122.0 (192.168.122.0)
         Netmask: 255.255.255.0 (255.255.255.0)
         Next Hop: 0.0.0.0 (0.0.0.0)
         Metric: 1
```

```
No.     Time           Source                Destination           Protocol Info
      130 85.561005     10.0.3.10             224.0.0.9             RIPv2    Response
 :
 (sniped)
 :
 Routing Information Protocol
     Command: Response (2)
     Version: RIPv2 (2)
     Routing Domain: 0
     IP Address: 10.0.1.0, Metric: 16
         Address Family: IP (2)
         Route Tag: 0
         IP Address: 10.0.1.0 (10.0.1.0)
         Netmask: 255.255.255.0 (255.255.255.0)
         Next Hop: 0.0.0.0 (0.0.0.0)
         Metric: 16

No.     Time           Source                Destination           Protocol Info
      131 87.560575     10.0.3.3              224.0.0.9             RIPv2    Response
 :
 (sniped)
 :
 Routing Information Protocol
     Command: Response (2)
     Version: RIPv2 (2)
     Routing Domain: 0
     IP Address: 10.0.1.0, Metric: 16
         Address Family: IP (2)
         Route Tag: 0
         IP Address: 10.0.1.0 (10.0.1.0)
         Netmask: 255.255.255.0 (255.255.255.0)
         Next Hop: 0.0.0.0 (0.0.0.0)
         Metric: 16

No.     Time           Source                Destination           Protocol Info
      132 87.561441     10.0.3.2              224.0.0.9             RIPv2    Response
 :
 (sniped)
 :
 Routing Information Protocol
     Command: Response (2)
     Version: RIPv2 (2)
     Routing Domain: 0
```

IP Address: 10.0.1.0, Metric: 16
Address Family: IP (2)
Route Tag: 0
IP Address: 10.0.1.0 (10.0.1.0)
Netmask: 255.255.255.0 (255.255.255.0)
Next Hop: 0.0.0.0 (0.0.0.0)
Metric: 16

No. Time Source Destination Protocol Info
133 89.802881 10.0.3.10 224.0.0.9 RIPv2 Response
:
(sniped)
:
Routing Information Protocol
Command: Response (2)
Version: RIPv2 (2)
Routing Domain: 0
IP Address: 10.0.1.0, Metric: 16
Address Family: IP (2)
Route Tag: 0
IP Address: 10.0.1.0 (10.0.1.0)
Netmask: 255.255.255.0 (255.255.255.0)
Next Hop: 0.0.0.0 (0.0.0.0)
Metric: 16
IP Address: 192.168.122.0, Metric: 1
Address Family: IP (2)
Route Tag: 0
IP Address: 192.168.122.0 (192.168.122.0)
Netmask: 255.255.255.0 (255.255.255.0)
Next Hop: 0.0.0.0 (0.0.0.0)
Metric: 1

No. Time Source Destination Protocol Info
135 90.717042 10.0.3.2 224.0.0.9 RIPv2 Response
:
(sniped)
:
Routing Information Protocol
Command: Response (2)
Version: RIPv2 (2)
Routing Domain: 0
IP Address: 10.0.1.0, Metric: 12
Address Family: IP (2)
Route Tag: 0

IP Address: 10.0.1.0 (10.0.1.0)
Netmask: 255.255.255.0 (255.255.255.0)
Next Hop: 0.0.0.0 (0.0.0.0)
Metric: 12

From looking at the "Routing Information Protocol" section of these six packets, we observed that the "Metric" value of network 10.0.1.0 started from 1, and it changed to 16. At the end, it changed again from 16 to 12.

PC3 was directly connected to network 10.0.1.0 at first, so the metric was 1 because there was one hop. Then the connection of PC3 interface 1 was disconnected from network 10.0.1.10. The metric therefore was changed to 16, which is the infinity metric for RIP protocol. When the routing table stabalized, PC3 had to change to go through Router1 in order to reach network 10.0.1.0. Router1 had a metric 10. This information was passed to PC3 as 10+1. The one was added because Router1 was one hop away from PC3. Also, the inner hop needed to be counted, so the metirc to reach network 10.0.1.0 changed to 10+1+1=12. This is the final stablized routing entry for network 10.0.1.0.

[Exercise 2C]
We changed the configurations of the routers to enable the triggered updates and then set the hold-down timer to a non-zero value. We used commands:

```
yourname# config t
yourname(config)# ip routing
yourname(config)# router rip
yourname(config-router)# version 2
yourname(config-router)# network 10.0.0.0
yourname(config-router)# timers basic 30 180 60 240
yourname(config-router)# flash-update-threshold 0
yourname(config-if)# end
```

We used command `show ip protocols` to have a look at the RIP timers. It was correctly set to 60. We also used traceroute from PC1 to PC4 to confirm that PC3's interface 1 was up. We observed that the packets from PC1 went through PC3 to reach the network 10.0.4.0.

Then, we repeated the previous exercise with this new setup. We saved the debug messages ex4cdebugpc1.txt. We still waited for about 3 minutes for the connection of PC3 to network 10.0.1.0 to reestablish.

[Exercise 3A-3B]
We connected the ethernet interfaces of the PCs and the routers. We used the

following commands to configure the IP addresses and the interfaces on the PCs:

on PC1: `ifconfig eth0 10.0.1.1/24`
on PC1: `ifconfig eth1 10.0.2.1/24`
on PC2: `ifconfig eth0 10.0.1.2/24`
on PC2: `ifconfig eth1 10.0.5.2/24`
on PC3: `ifconfig eth0 10.0.3.4/24`
on PC3: `ifconfig eth1 10.0.4.4/24`
on PC4: `ifconfig eth0 10.0.6.7/24`
on PC4: `ifconfig eth1 10.0.7.7/24`

We used the following commands to turn on the OSPF protocol and configure the IP addresses and interfaces on the routers:

<Router1>

```
on PC1: `minicom`
username: cisco
password: cisco
yourname# config t
yourname(config)# no ip routing
yourname(config)# ip routing
yourname(config)# no router rip
yourname(config)# router ospf 1
yourname(config-router)# network 10.0.0.0 255.255.255 area 1
yourname(config-router)# interface fa0/0
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.3.3 255.255.255.0
yourname(config-if)# interface fa0/1
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.2.3 255.255.255.0
yourname(config-if)# end
yourname# clear ip route *
```

<Router2>

```
yourname(config)# router ospf 1
yourname(config-router)# network 10.0.0.0 255.255.255 area 1
yourname(config-router)# interface fa0/0
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.4.5 255.255.255.0
yourname(config-if)# interface fa0/1
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.2.5 255.255.255.0
yourname(config-if)# end
yourname# clear ip route *
```

<Router3>

```
yourname(config)# router ospf 1
yourname(config-router)# network 10.0.0.0 255.255.255 area 1
yourname(config-router)# interface fa0/0
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.5.6 255.255.255.0
yourname(config-if)# interface fa0/1
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.6.6 255.255.255.0
yourname(config-if)# end
yourname# clear ip route *
```

<Router4>

```
yourname(config)# router ospf 1
yourname(config-router)# network 10.0.0.0 255.255.255 area 1
yourname(config-router)# interface fa0/0
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.5.8 255.255.255.0
yourname(config-if)# interface fa0/1
yourname(config-if)# no shutdown
yourname(config-if)# ip address 10.0.7.8 255.255.255.0
yourname(config-if)# end
yourname# clear ip route *
```

We configured the PCs to run OSPF using the Zebra package. We first enabled IP forwarding on the PCs and added the Zebra directory to the path. The commands we used were as follows:

```
on PC#: `echo "1" > /proc/sys/net/ipv4/ip_forward`
on PC#: `export PATH=/etc/rc.d/init.d/:$PATH`
on PC#: `zebra start`
on PC#: `ripd stop`
on PC#: `ripd status`
on PC#: `zebra restart`
on PC#: `ospfd start`
on PC#: `telnet localhost 2604`
```

Since this was the begining of the lab and we didn't run ripd on the machines before this exercise, the commands `ripd stop` was not needed. We further configured the PCs to run ospfd. We used commands:

<PC1>

```
ospf> enable
```

```
ospf# config t
ospf(config)# router ospf
ospf(config-router)# network 10.0.0.0/8 area 1
ospf(config-router)# router-id 10.0.1.1
ospf(config-router)# no passive-interface eth0
ospf(config-router)# no passive-interface eth1
ospf(config-if)# end
ospf# exit
```

<PC2>

```
ospf> enable
ospf# config t
ospf(config)# router ospf
ospf(config-router)# network 10.0.0.0/8 area 1
ospf(config-router)# router-id 10.0.1.2
ospf(config-router)# no passive-interface eth0
ospf(config-router)# no passive-interface eth1
ospf(config-if)# end
ospf# exit
```

<PC3>

```
ospf> enable
ospf# config t
ospf(config)# router ospf
ospf(config-router)# network 10.0.0.0/8 area 1
ospf(config-router)# router-id 10.0.3.4
ospf(config-router)# no passive-interface eth0
ospf(config-router)# no passive-interface eth1
ospf(config-if)# end
ospf# exit
```

<PC4>

```
ospf> enable
ospf# config t
ospf(config)# router ospf
ospf(config-router)# network 10.0.0.0/8 area 1
ospf(config-router)# router-id 10.0.6.7
ospf(config-router)# no passive-interface eth0
ospf(config-router)# no passive-interface eth1
ospf(config-if)# end
ospf# exit
```

We used traceroute and ping commands to verify the configuration. The commands did not go through at first. We added 'no shutdown' to configure the routers'

interfaces and issued the commands again. This time, they were successful. We also used command `netstat -rn` on the PCs and command `show ip route` on the routers to display the routing tables. The hosts and routers all had routing information of the other networks in the system.

[Exercise 3C]
We started the wireshark on PC1 and set a display filter to display only OSPF packets. We issued a traceroute from PC3 to PC4 using `traceroute 10.0.7.7`. We observed that the packets went from PC3 to Router 2 to PC1 to PC2 to Router4 and reached PC4.

10.0.4.4 10.0.4.5 10.0.2.1 10.0.1.2 10.0.5.8 10.0.7.7

We started a ping command from PC3 to PC4 `ping 10.0.7.7` with no time limitation. We then disconnected the interface 0 on Router4. This disconnection removed Router4 from the picture.
[Exercise 4]
We reconfigured the PCs and routers to assgin different area numbers to these network entities. We first restarted ospfd using `zebra restart` and `ospfd restart`. We then used the following commands to configure the area numbers:

<PC3>
```
ospf> enable
ospf# config t
ospf(config)# router ospf
ospf(config-router)#  router-id 10.0.3.4
ospf(config-router)# network 10.0.3.0/24 area 1
ospf(config-router)# network 10.0.4.0/24 area 1
ospf(config-if)# end
ospf# exit
```

<PC4>
```
ospf> enable
ospf# config t
ospf(config)# router ospf
ospf(config-router)#  router-id 10.0.6.7
ospf(config-router)# network 10.0.6.0/24 area 2
ospf(config-router)# network 10.0.7.0/24 area 2
ospf(config-if)# end
ospf# exit
```

<PC1>
```
ospf> enable
```

```
ospf# config t
ospf(config)# router ospf
ospf(config-router)#    router-id 10.0.1.1
ospf(config-router)# network 10.0.2.0/24 area 1
ospf(config-router)# network 10.0.1.0/24 area 0
ospf(config-if)# end
ospf# exit
```

<PC2>

```
ospf> enable
ospf# config t
ospf(config)# router ospf
ospf(config-router)#    router-id 10.0.1.2
ospf(config-router)# network 10.0.5.0/24 area 2
ospf(config-router)# network 10.0.1.0/24 area 0
ospf(config-if)# end
ospf# exit
```

<Router1>

```
yourname# config t
yourname(config)# no router ospf 1
yourname(config)# router ospf 1
yourname(config-router)# network 10.0.3.0 0.0.0.255 area 1
yourname(config-router)# network 10.0.2.0 0.0.0.255 area 1
yourname(config-router)# end
yourname # clear ip ospf 1 process
```

<Router2>

```
yourname# config t
yourname(config)# no router ospf 1
yourname(config)# router ospf 1
yourname(config-router)# network 10.0.2.0 0.0.0.255 area 1
yourname(config-router)# network 10.0.4.0 0.0.0.255 area 1
yourname(config-router)# end
yourname # clear ip ospf 1 process
```

<Router3>

```
yourname# config t
yourname(config)# no router ospf 1
yourname(config)# router ospf 1
yourname(config-router)# network 10.0.5.0 0.0.0.255 area 2
yourname(config-router)# network 10.0.6.0 0.0.0.255 area 2
yourname(config-router)# end
yourname # clear ip ospf 1 process
```

<Router4>

```
yourname# config t
yourname(config)# no router ospf 1
yourname(config)# router ospf 1
yourname(config-router)# network 10.0.5.0 0.0.0.255 area 2
yourname(config-router)# network 10.0.7.0 0.0.0.255 area 2
yourname(config-router)# end
yourname # clear ip ospf 1 process
```

We displayed the link-stated databases using command `show ip ospf datasase` on the routers and command `show ip ospf database` on the PCs. We saved the link state databases of the PCs and router as ex6-4PC1.txt, ex6-4PC2.txt, ex6-4PC3.txt, ex6-4PC4.txt, and ex6-4routers.txt.

Exp5 Transport Layer

1.Introduction

In this experiment, you will learn the operation of the two transport layer protocols: UDP and TCP. UDP is a simple protocol for exchanging messages. TCP is a connection oriented protocol which ensures reliable delivery of data.

2.Assignment

Pre-lab

Refresh the use of the trafclient and trafserver from the previous experiments. These will be used to generate pseudo traffic in this experiment. Read about UDP and TCP from the textbook. Read about the TCP congestion control.

[Question 1]

Explain the role of port numbers in TCP and UDP

To use different port numbers for different protocols allows the host to receive more than one protocol transmissions. The receiver tells the sender what port number it is using for this certain kind of protocol, the sender would sent to that port. Also specifying the port numbers allow the host to decide which transmissions to listen to but not the others.

[Question 2]

Provide the syntax of the ttcp command for both the sender and receiver which executes the following scenario:

A TCP server has IP address 10.0.2.6 and a TCP client has IP address 10.0.2.7. The TCP server is waiting on port number 2222 for a connection request The client connects to the server and transmits 2000 bytes to the server, which are sent as four write operations of 500 bytes each.

on receiverPC (server)#: ttcp -rs -l2000 -n4 -p 2222
on senderPC (client)#: ttcp -ts -l2000 -n4 -p2222 -D 10.0.2.6

[Question 3]

Answer the followig questions on Path MTU Discovery:

a. How does TCP decide the maximum size of a TCP segment?

This Maximum Segment Size (MSS) announcement is sent from the data receiver to the data sender and says "I can accept TCP segments up to size X". The size (X) may be larger or smaller than the default.

b. How does UDP decide the maximum size of a UDP datagram?

UDP provides a message-oriented interface. Each message is sent as a single UDP segment, which means that data boundaries are preserved. However, this also means that the maximum size of a UDP segment depends on the maximum size of an IP datagram, which is 576 bytes. Allowing large UDP segments can cause problems. Processes sending large segments can result in IP fragmentation.

c. What is the ICMP error generated by a router when it needs to fragment a datagram with the DF bit set? Is the MTU of the interface that caused the fragmentation also returned?

The basic procedure of path MTU discovery is simple--send the largest packet the sender can, and if it won't fit through, some link get back a notification saying what size will fit. The notifications arrive as ICMP packets known as "fragmentation needed" ICMPs. The notifications are requested by setting the "do not fragment" (DF) bit in packets that are sent out.

So to answer the question, ICMP error "fragment needed" messages are generated. The MTU interface that caused the fragmentation would return and tell what size will fit.

d. Explain why a TCP connection over an Ethernet segment never runs into problems with fragmentation.

Ethernet MTU is 1500, and MSS is 1460. TCP runs over IP and IP fragment is smaller than ethernet fragment size.

[Question 4]
Assume a TCP sender receives an ackowledgment (ACK)--that is, a TCP segment with the ACK flag set--in which the acknowledgment number is set to 34567 and te window size is set to 2048. Which sequence number can the sender transmit.

Window (16 bits)--the size of the receive window, which specifies the number of bytes (beyond the sequence number in the acknowledgment field) that the receiver is currently willing to receive. The sequence number can the sender transmit is 34567 - 2048 = 32519.

[Question 5]
Describe the following heuristics in TCP and explan why thay are used.

a. Nagle's algorithm: Nagle's algorithm works by coalescing a number of small outgoing messages, and sending them all at once. Specifically, as long as there

is a sent packet for which the sender has received no acknowledgment, the sender should keep buffering its output until it has a full packet's worth of output, so that output can be sent all at once.

b. Karn's algorthm: Karn's Algorithm requires not using retransmitted segments when updating the round trip estimate. Round trip time estimation is based only on unambiguous acknowledgments, which are acknowledgments for segments that were sent only once.

A solution to the problem that the ack messages may be for the previous transmission, is to incorporate transmission timeouts with a timer backoff strategy. The timer backoff strategy computes an initial timeout. If the timer expires and causes a retransmission, TCP increases the timeout generally by a factor of 2.

[Question 6]
Anwer the following question about TCP acknowledgments:

a. What is a delayed acknowledgment?

It is a TCP preformance problem. The server waits for the client to ACK the segment before responding. Delayed acknowledgment means that the client fail to do so for up to 200ms.

b. What is a piggybacked acknowledgment?

To reduce the number of segments, we can combine the 2nd and 3rd segments in a delayed acknowledgement. In this case, each end waits for about 200 milliseconds to see if there is any data going in the same direction as acknowledgement. If it finds one, the acknowledgement can be piggybacked with the data.

[Question 7]
Describe how the retransmission timeout (RTO) value is determined in TCP

TCP provides a reliable transport layer. One of the ways it provides reliability is for each end to acknowledge the data it receives from. the other end. But data segments and acknowledgments can get lost. TCP handles this by setting a timeout when it sends data, and if the data isn't acknowledged when the timeout expires, it retransmits the data. A critical element of any implementation is the timeout and retransmission strategy.

Fundamental to TCP's timeout and retransmission is the measurement of the

round-trip time (RTT) experienced on a given connection. We expect this can change over time, as routes might change and as network traffic changes, and TCP should track these changes and modify its timeout accordingly.

[Question 8]
Answer the following questions on TCP flow control and congestion avoidance in TCP

a. Describe the sliding window flow control mechanism used in TCP

Sliding Window Flow Control is a method of flow control in which a receiver gives the transmitter permission to transmit data until a window is full. When the window is full, the transmitter must stop transmitting until the receiver advertises a larger window. TCP uses this method of flow control.

b. Describe the concepts of slow start and congestion avoidance in TCP

Congestion window starts small, at 1 segment size. Each time a transmitted segment is acknowledged, the congestion window is increased by one maximum segment size. Congestion window size grows exponentially (i.e. it keeps on doubling). Packet losses indicate congestion. Packet losses are determined by using timers at the sender. When a timeout occurs, the congestion window is reduced to one maximum segment size and everything starts over.

Congestion avoidance in TCP is to establish a threshold at which the rate increase is linear instead of exponential to improve efficiency.

c. Explain the concept of fast retransmit and fast recovery in TCP.

Fast retransmis is another enhancement to TCP congestion control. The idea is when sender sees 3 duplicate ACKs, it assumes something went wrong. The packet is immediately retransmitted instead of waiting for it to timeout.
Fast recovery is another enhancement to TCP congestion control. The idea is don't do a slow start after a fast retransmit.